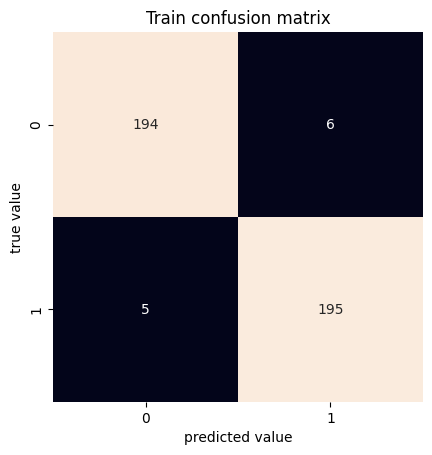
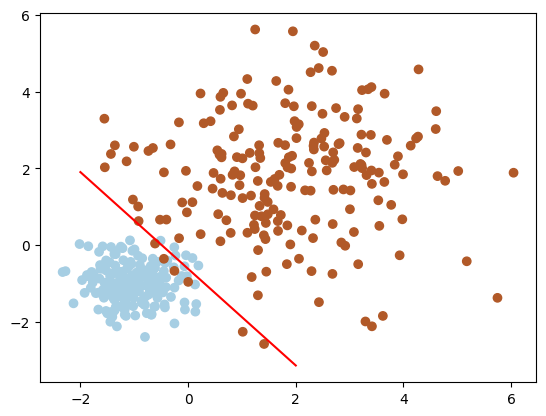
**MACHINE LEARNING FROM DATA**

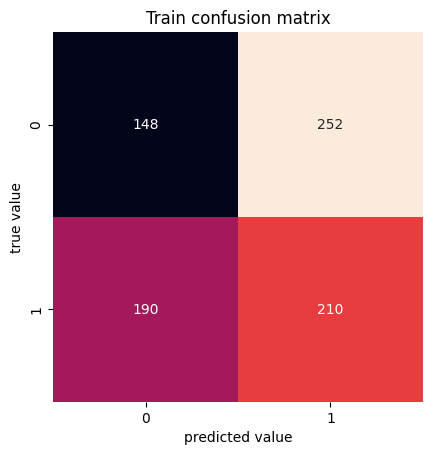
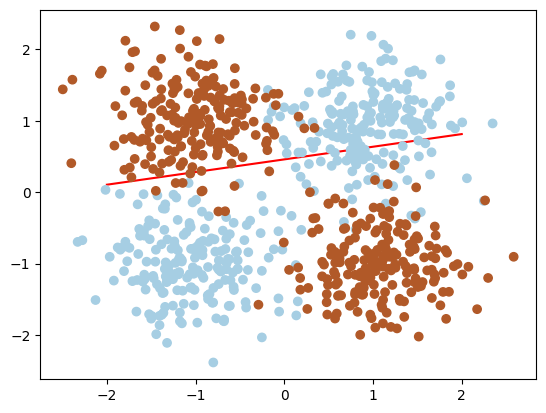
**Report: Lab Session 7 – Neural Networks**

**Q1**. Which are the default parameters used by the Perceptron class? (check scikit-learn documentation). Compare the performance of the Perceptron on the two toy examples (linearly and non-linearly separable). Compare the performance of the Perceptron when using the original 2D features and features augmented by interaction

**LINEARLY SEPARABLE (train error: 0.027500)**

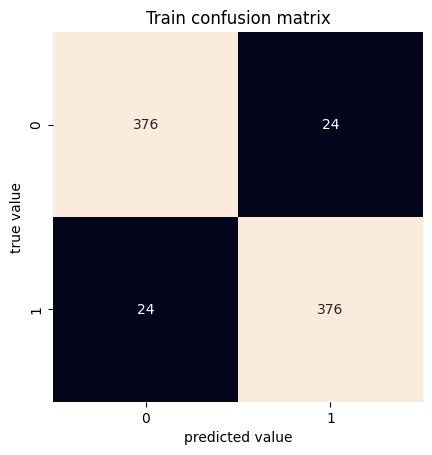
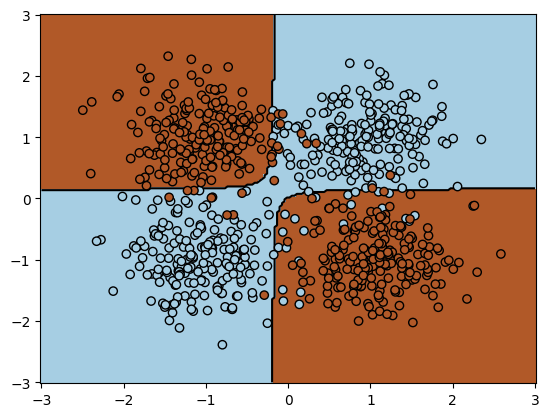


**NON-LINEARLY SEPARABLE (train error: 0.552500)**



For the non-linearly separable example, we can see that the error is huge and the confusion matrix also shows that this classifier is very bad for this problem.

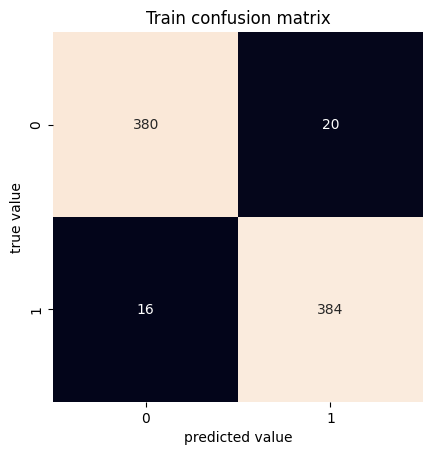
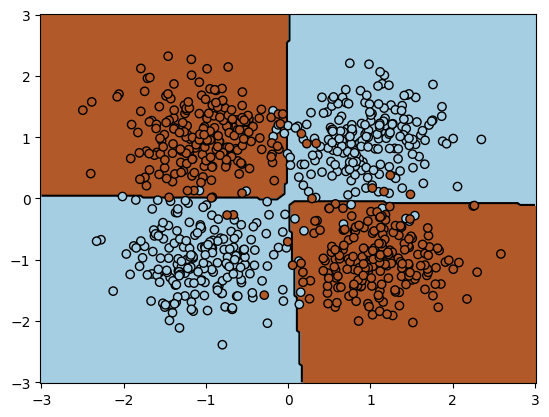
**NON-LINEARLY SEPARABLE WITH INTERACTION FEATURE (train error: 0.060000)**



For this problem, we can see that when adding a third feature by interaction of the other two features, the classifier works much better and reduces the error to a very low percentage.

**Q2**. Compare the performance of the Perceptron with 3D features (with interaction) and the Multi-Layer perceptron with 2D features (you can try to improve the performance by varying some hyperparameters).

**Multi-Layer perceptron with 2D (train error: 0.045000)**

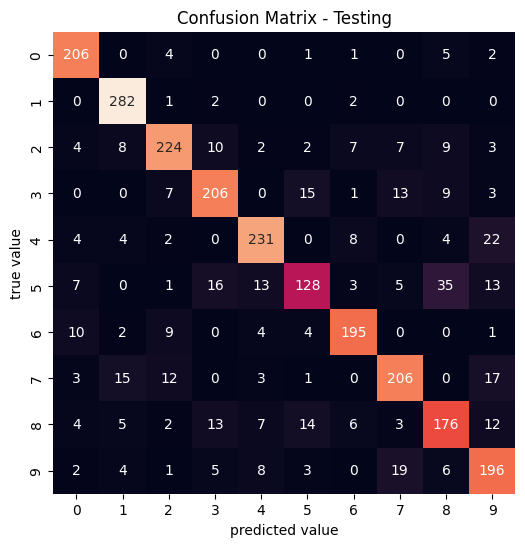
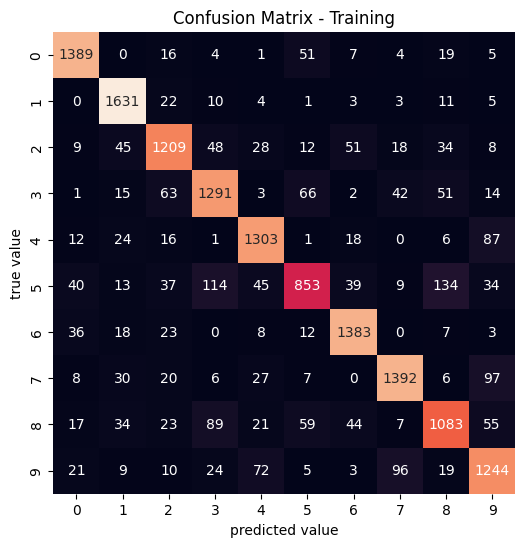


When comparing the MLP to the perceptron we can see that the classifier behaves very similarly. The error is a bit lower in the MLP compared to the perceptron. The decision regions are very similar, but the decision boundary for the MLP is a bit more complex than the one for the perceptron, allowing a lower error rate.

If we use the same type of classifier (MLP), but we add more layers or more hidden units per layer, we can see that the classifier lowers the error even more. But we can eventually be overfitting too much depending on the complexity of the model.

**Q3**: For the MNIST task, copy the global accuracy and the confusion matrix for the training and test set and analyze the results.

**MLP Classifier MNIST**



train accuracy: 0.8518666666666667

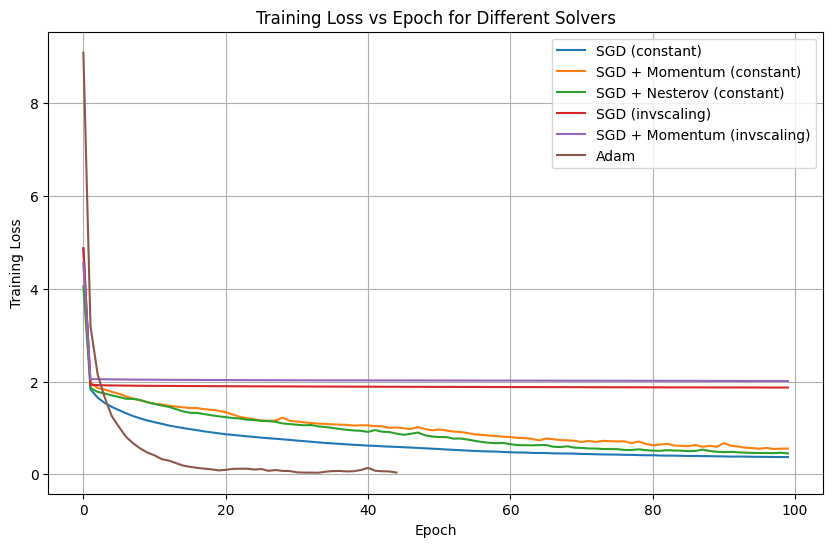
train error: 0.14813333333333334

test accuracy: 0.82

test error: 0.18000000000000005

We see that the test error is slightly larger than the train error and also that the classifier works quite well, however it has an error of almost 20%.

**Q4**. For the MNIST task, analyze and compare the methods based on the loss curves.



We can see that when using invscaling the classifiers converge earlier however the loss value is much greater so we can assume that the resulting classifiers will be much worse than when using sgd or adam.

Nesterov momentum seems to be better than constant momentum and base sgd, as it lowers the loss value faster and in the invscaling case it converges in a lower value for the loss.

Finally adam clearly is the best classifier of all as it lowers much quicker and converges at the lowest point of all regarding the loss value.

**Q5**. Analyze the results provided by grid\_search.cv\_results\_

Mention if you find significant differences in performances for some of the hyperparameters.

We can see that the models with hidden layers (100,100) in general terms have a worst performance than the ones with 100. The models performing best are using a learning rate of 0.010 instead than 0.001.

The models performing better uses relu activation too

**Q6**. Classify the training and test sets with the best hyperparameters. Compute the classification reports, accuracy, error and confusion matrices for the training and test sets. Discuss the results.

The best parameters are:

{'mlp\_\_activation': 'relu', 'mlp\_\_alpha': 0.001, 'mlp\_\_batch\_size': 16, 'mlp\_\_hidden\_layer\_sizes': (100,), 'mlp\_\_learning\_rate': 'constant', 'mlp\_\_learning\_rate\_init': 0.001}

We have a really good accuracy value 95% for the test set, this is showing that the model is generalizing well. The precision, recall, and f1-score are consistent across the classes, this is showing that the results are not biased the f1-score value is pretty good in all cases with a min. value of 0.94